Remote Commanding Documentation

5 October 2006

1 Remote commanding applications

Overview of version 1.1

The ITOS remote commanding utility consists of three java applications:

1. SendCmd runs on remote computers, accepts commands from operators and sends them to RecvCmd.
2. RecvCmd runs on the main ITOS computer, accepts commands from SendCmd applications and passes them to ITOS.
3. RelayCmd can be used if the main ITOS computer is separated from SendCmd clients by a firewall. RelayCmd runs on a gateway computer and relays commands from the open internet to a closed network. Several RelayCmd applications can be linked to one another if that is necessary.

Scripts runsend, runrecv and runrelay can be used to run the applications. They set $CLASSPATH and may set environment variables for the applications.
The behavior of all three applications can be modified extensively with startup arguments. For example, commands can be encrypted or sent plaintext. RecvCmd can send commands to ITOS immediately or can hold commands until a local operator approves them. The local operator can be given or denied the ability to modify commands.

System Requirements
Obtaining and installing the applications
Example argument files
Encryption/Decryption operations

2 System requirements

1. All three applications require java 1.1.6 or later. There have been a few problems with java 1.1.7, so that version should be avoided. Java can be downloaded from or

   NOTE: These applications do not run correctly under the Kaffe virtual machine. Use the jdk virtual machines mentioned above.

2. The tarfile without gpg is 216k.

3. Tarfiles containing gpg range from 699k (Linux version) to 1284k (Solaris pc version).

4. The remote commanding directory, after expanding and first use, is 4 to 5 times the size of the tar file.

3 Installing ITOS Remote Commanding

1. If you have the ITOS installation on your machine, the tar files are located in a subdirectory of the ITOS install directory named '`pkgs`'. If you don't have our source files, contact the ITOS team at itos@itos.gsfc.nasa.gov to get the address for ftp access to the tar files. **WARNING – These files are part of ITOS and still carry the same export restrictions. Before you can distribute the `RERCMD` package to anyone you must determine that it does not violate US law. If you are not sure, contact us at itos@itos.gsfc.nasa.gov for information.**

2. The following files are available and contain all software needed to run the remote commanding applications, including an executable for gpg encryption built for the operating system indicated.
   - '`remcmd.i386-FreeBSD-3.2.tar.gz`'
   - '`remcmd.i386-SunOS-5.7.tar.gz`'
   - '`remcmd.sparc-SunOS-5.7.tar.gz`'
   - '`remcmd.i386-Linux-2.2.12-20.tar.gz`'

   '`remcmd.nogpg.tar.gz`'
   Contains everything needed to run the remote commanding applications except gpg encryption software. Use this file if you will not use encryption or you'll install gpg separately. You'll have to do that if your operating system is not listed above.
   Gpg is free and can be downloaded from
3. Put the tar file in the directory where remcmd will be installed. Any directory can be used, as long as users of remcmd have write permission in the directory.

4. Unpack the tar file:
   ```
   tar zvxpmf xxx.tar.gz
   ```
   That will create:
   - Scripts runsend, runrecv, runrelay and rungpg
   - Argument files sendargs, recvargs, relayargs
   - Installation script INSTALL
   - Directory classes, containing the application java class files.
   - Directory gpg, containing a set of encryption keys for testing and the gpg executable. (The executable is not included in remcmd.nogpg.tar.gz.)

5. Run INSTALL by typing `./INSTALL`. That puts the installation directory into several argument files and shell scripts. There is no UNINSTALL. If this directory is moved in the future, just rerun INSTALL.

6. If you will be running SendCmd, edit argument file '`sendargs`'. The first argument is currently

   ```
   -RecvCmd         <itos host>      6100
   ```

   Change "<itos host>" to the name or IP address of the host where commands will be sent.
7. If you will be running RecvCmd, edit argument file '`recvargs`'. The first argument is currently

   ```
   -arg ItosComm stolfifo <fifo path>
   ```

   Change "<fifo path>" to the complete path of the STOL fifo. You can find that path with the STOL command:

   ```
   system "echo $ITOS_STOLFIFO"
   ```

   The following message will be displayed in any event window displaying STOL events:

   ```
   STOL_MSG: SYSTEM: <path>
   ```

   Note that this message is printed before the STOL_ECHO message echoing your input.
8. If you will be using RelayCmd, edit the argument file '`recvargs`' by un-commenting the last four arguments in the file, and replacing "<host name>" in the last argument with the host where RelayCmd is running.

Go to SendCmd, RecvCmd or RelayCmd for information on invoking and running the application you are interested in.

4 The SendCmd application

SendCmd is a java application that runs on a host remote from the ITOS host that allows an operator to enter ITOS STOL commands and send them to the ITOS host. It displays messages when commands are:

- encrypted
- sent to RecvCmd
- decrypted by RecvCmd
- accepted by a RecvCmd operator and sent to ITOS
- rejected by a RecvCmd operator

SendCmd also:

- Lets the operator specify an expiration interval for a command. The command must be accepted within that time interval.
- Lets the operator include a message to the RecvCmd operator with the command.
- Displays all messages related to a single command in a window.
- Lets the operator query RecvCmd about the current status of any command.
- Lets the operator query RecvCmd about whether ITOS is currently running.

Invoking SendCmd
Using SendCmd
Using the SendCmd menu interface
SendCmd command-line arguments
Example argument files

4.1 Invoking SendCmd

After installation, SendCmd can be started by invoking '`runsend`', usually with no arguments.

'`runsend`' is a convenience Bourne shell script that sets $CLASSPATH, passes the argument file '`sendargs`' to SendCmd and runs it in the background.

SendCmd operation can be modified in two ways:

1. By editing the argument file '`sendargs`'. This is probably the easier method and is described below.
2. By including arguments in the call to '`runsend`'. Any number of arguments can be given. Type "runsend -help" or "runsend ?" for a list of arguments recognized. Those arguments are explained in more detail in the command-line arguments section.

The rest of this page describes the contents of '`sendargs`'. There are also example argument files.

The argument file '`sendargs`' contains the command-line arguments most likely to be useful. Any text following '#' is a comment. The arguments are:

1. -RecvCmd <itos host> 6100
   This argument is required. As part of installation, "<itos host>" must be replaced by the host name or IP address of the machine where RecvCmd or RelayCmd is running. "6100" is the default server port and usually does not need to be changed.
2. -arg GPG1 local remote-user
   Specifies the user whose private key will be used to sign outgoing messages if gpg is used for encryption. If GPG1 is not used for encryption, this argument should be commented-out. When new keys are generated for gpg, "remote-user" should be changed to the new name. If the user name contains blanks, it must be enclosed in single or double quotes.
3. -arg GPG1 remote itos-user
   Specifies the user whose public key will be used to encrypt outgoing messages. This should be the same user as RecvCmd's '`-arg GPG1 local`' user. If GPG1 is not used for encryption, this argument should be commented out. When new keys are generated for gpg, "itos-user" should be changed to the new name.
4. -arg GPG1 gpgpath <path>
   Specifies the complete path the the gpg executable. The INSTALL script replaces "<path>" with the path.
5. -arg GPG1 keydir <directory>
   Specifies the directory where gpg will look for keys. The INSTALL script replaces "<directory>" with the directory.
6. #-arg GPG1 sign false
   If this argument is uncommented, outgoing messages are not signed.
7. #-arg GPG1 promptAll true
   By default, an operator is required to enter the local user's passphrase only once during an invocation of SendCmd. If this argument is uncommented, The operator will be queried for the passphrase every time a message is received and, if messages are signed, every time a message is sent.
8. #-arg GPG1 tempdir <directory>
   Specifies the directory where temporary files resulting from encryption and decryption are put. The default is "/tmp/".
9. #-localPort <port> Specifies the server port opened on the local host to receive messages from RecvCmd. The default is 6100. If this port is changed while RecvCmd is processing commands from this SendCmd, RecvCmd will send messages regarding those commands to the wrong port. However, SendCmd can send pings to get information on those commands.
10. #-logdir <directory>
    Specifies where log files are put. The default is "/tmp/". To change that, uncomment this argument and replace "<directory>" with a directory path. Log files are never removed or truncated, so it is up to system administrators to delete old files.
    See "-Log" below for a way to stop log file creation.
11. #-logname <filename>
    Specifies the name of the log file. The default is '`SendCmd-mm-dd.log`' where "mm" is the number of the month, "dd" is the date.

12. #-Log NoLog
    Uncommenting this argument will stop any log file from being created.
13. #-Decryptor NoDecryption
    #-Encryptor NoEncryption
    Uncommenting these arguments results in commands being sent from SendCmd to RecvCmd unencrypted. This is the preferred way to send commands on a closed network.
    If these arguments are uncommented, all arguments above that mention GPG1 should be commented out.
14. #-arg SendCmdDB delete <days>
    Specifies how long a command stays in the database. The default is two days.
15. #-arg SendCmdDB dbfile <path>
    Specifies where the database file is kept. The default is '`/tmp/remcmd.senddb`'. The database file allows SendCmd to keep track of commands between invocations.
16. #-CmdBuilder <class>
    Specifies the command builder class. With the default command builder commands are typed into an input field. A menu-driven builder is also available.
17. #-SendCmdDB <class> Specifies the database class. There is only one database class at the moment.
18. #-RecvComm <class>
    Specifies the class that handles communications with RecvCmd. There is only one such class at the moment.

4.2 Using SendCmd

The SendCmd frame contains three buttons and two scrolling windows.

The bottom window is an event window that displays timestamped messages when noteworthy events occur. These same messages are written to the log file if a log file is being created.

To create a command, click the "Send a command" button. (If you are using the menu-driven interface a menu will be put on top of the window described here.)

A window appears with the title "Send a command" and several input fields:

- Type the STOL command into the top input field.
- Type an abbreviation of the command into the "Command ID" field. This abbreviation will be used in information messages about the command. If no abbreviation is entered, the system will create a name with the format "cmdN" where "N" is a number.

  This command ID may also be used when you ping RecvCmd to get information on a command's status. If you have sent the same command more than once, the command ID is used to decide which instance is being pinged.
- The "Expiration" field lets you specify how long the command can stay in RecvCmd before it is accepted. The default is normally three hours, but that can be changed with a startup argument to RecvCmd. Clicking on the button next to "Expiration", brings up a menu with the following choices:

  - Default, described already.
  - No expiration.
  - Explicitly enter the expiration interval in days and hours.
- The "Sender ID" field is optional, and can contain the sending operator's name. If entered, it gives information to the RecvCmd operator and is used in identifying the correct command in ping queries.
- The "comment" field is also optional. If entered, any comment is visible to the RecvCmd operator.
- At the top of the window is a "Clear" button that clears all input fields and resets the expiration selector to the default.

After entering command information, click the "Send" button at the bottom of the window. The window will disappear and the message

```
Processing <command ID>...
```

is printed in the event window.

You may be prompted for a passphrase so the message can be signed. This depends on whether the argument file specified that messages are being encrypted, are being signed, and whether the passphrase is required for every message or only the first.

If the passphrase is mis-typed, a message like

```
gpg: skipped 'remote-user': bad passphrase
```

is printed in the event window and the passphrase dialog is re-displayed. If the passphrase is mis-typed three times, the command is discarded.

The message

```
Encrypted <command ID>
```

is printed in the event window after successful encryption and the command is sent to RecvCmd.

An error message like

```
FAILED to send cmd5: Can't connect to <host specification>
```

probably means RecvCmd is not running at the moment.

The ID of the new command is displayed at the top of the upper scrolled window of the SendCmd frame along with the time it was created. The background color used with the command gives some information about its status. A green background means the command has been sent to RecvCmd but has not been accepted or rejected yet. A light gray background means the command has been accepted. A dark gray background means the command could not be sent to RecvCmd or that the command has been rejected by an operator at RecvCmd.

Clicking on any command in this upper list brings up a window containing the command's text, ID, sender and all event messages that relate to the command. The window also contains three buttons:

Ping this command
: sends a request to RecvCmd for information about the command. The response from RecvCmd is printed in the event window and in this command-information

window. This can be useful if SendCmd is shut down for a while and may have missed messages from RecvCmd.

Delete this command
: removes the command from the list and from the database. By default, commands are removed from the list and database two days after creation. This button allows uninteresting commands to be deleted early.

Cancel
: simply hides the window.

The button "Check connection to ITOS" in the SendCmd frame sends a request to RecvCmd about the status of its connection to ITOS. Clicking that button commonly results in one of the following series of messages in the event window:

1. If RecvCmd and ITOS are both running:

```
Processing ping...
2000-045-20:39:23 Encrypted ping
2000-045-20:39:23 Ping successfully sent to RecvCmd.
2000-045-20:39:32 Reply to Ping: Connection to ITOS is OK.
```

2. If RecvCmd is running but ITOS is not:

```
Processing ping...
2000-045-20:44:52 Encrypted ping
2000-045-20:44:52 Ping successfully sent to RecvCmd.
2000-045-20:45:06 Reply to Ping: Cannot send to ITOS: Can't open ...
```

3. If RecvCmd is not running:

```
Processing ping...
2000-045-20:46:18 Encrypted ping
2000-045-20:46:18 ERROR sending ping to RecvCmd: Can't connect to ...
```

4.3 SendCmd menu-driven command builder

When the menu-driven command builder is in use, clicking on "Send a command" displays a window with the title "Select a command". Its largest panel contains buttons for commands along with command descriptions.

To send a command:

1. Click on a command button.
2. If the command has submnemonics, a window will appear with a panel for each submnemonic. Many submenmonics have default values, and "DEFAULT" is displayed in their entry fields. They can be modified if you like.
3. Submnemonics without default values have blank entry fields and you must enter a value in those fields.
4. When all submenmonics have been set, click "OK". The menu will disappear and the command you've created is displayed in the usual text command builder. You can now modify the command's expiration, enter your name and comments.
5. Click "Send" at the bottom of this window to send the command to ITOS.

At startup, the large panel displays a button for every command. That list of commands can be shortened by clicking on one of the subsystem buttons in the panel to the right. Only commands from the indicated subsystem will be displayed.

The list can also be shortened by entering a regular expression in the field at the top of the window. For example, typing "`time`" and clicking "Search" (or pressing return) displays all commands that contain "time". Similarly, typing "`in.*time`" lists all commands that contain "in" followed later by "time".

The "All Cmds" button at the top of the menu frame restores the whole list after it has been shortened.

4.4 Setting up the menu-driven command builder

1. Get ITOS dbx record files describing the commands that will appear in the menu. It doesn't matter where you put them in the file system. You'll give their paths in sendargs as described below.
2. Edit file sendargs:
   1. Uncomment the line "-CmdBuilder MenuBuilder".
   2. Uncomment the line "-arg MenuBuilder mneformat <format>" and change "<format>" to "sdb".
   3. Uncomment the line "-arg MenuBuilder mnefile <path>" and change "<path>" to the pathname of a dbx file. There can be any number of these lines, each giving a different path. Path can also be a directory. SendCmd will search that directory and all subdirectories for files with the extension ".sdb" and take command information from them.

4.5 SendCmd command-line arguments

It is often more convienient to edit the argument file '`sendargs`' than to type command-line arguments.

One argument is **always required** by SendCmd:

```
-RecvCmd <addr> <port>
```

The arguments specify the TCP/IP socket to which commands will be sent. It should be the server port opened by RecvCmd or RelayCmd.

The following arguments are **usually required** by SendCmd when GPG is used for encryption and decryption:

```
-arg GPG1 local <name>
```

Specifies the name used to sign outgoing messages and to decrypt incoming messages. Default is "itos-user" and is probably not correct for SendCmd.

```
-arg GPG1 remote <name>
```

Specifies the user name used to encrypt outgoing messages. Default is "remote-user" and is probably not correct for SendCmd.

Optional arguments:

-argfile <path>
: Reads arguments from file <path>. Arguments in the file are treated like those typed on the command line. Line breaks can appear anywhere in argument files except in the middle of words. Blank lines and all text following '#' on a line are ignored.

-Encryptor <class>
: Specifies the Encryptor class, which encrypts and signs messages before sending them to RecvCmd. The default is class "GPG1", which uses the Gnu Privacy Guard for encryption. The other class currently available is "NoEncryption", which does not encrypt messages. The encryptor class should match the decryptor class used by RecvCmd.

-Decryptor <class>
: Specifies the Decryptor class, which decrypts messages from RecvCmd. The default is class "GPG1". The other decryptor class currently available is "NoDecryption", which assumes messages do not need to be decrypted.

-localPort <port>
: Specifies the server port opened by SendCmd to receive data from RecvCmd. Default is port 6100.

-Log <class>
: Specifies the Log class, which writes the log file. The default is class "Log1". The other Log class currently available is class "NoLog", which does not write a log file.

-logdir <directory>
: Specifies where the log file is put. The default is "/tmp/".

-logname <filename>
: Specifies the log file name. The default is constructed from the date when SendCmd is started and is something like "SendCmd-3-24.log".

-arg SendDB1 dbfile <path>
: Specifies where the database file is put. The default is "/tmp/remcmd.senddb". The database file allows SendCmd to display information about commands constructed during previous sessions.

-arg SendDB1 delete <days>
: Specifies how long a command remains in the database. The default is 2 days.

The following commands can be used when GPG1 is used for encryption and decryption.

-arg GPG1 sign <true/false>
: Specifies whether outgoing messages are signed. Default is "true".

-arg GPG1 promptAll <true/false>
: Specifies whether the operator is prompted for the local user's passphrase every time an incoming message is decrypted or an outgoing message signed. If false, the operator is prompted only the first time the passphrase is needed. Default is "false"

-arg GPG1 tempdir <path>
: Specifies where temporary files are put. Default is '`/tmp/`'.

-arg GPG1 gpgpath <path>
: Specifies the path to the gpg executable. The default is "gpg", so this argument is necessary only if gpg's directory is not in the $PATH environment variable.

-arg GPG1 keydir <path>
: Specifies the directory containing GPG encryption keys. The default is '`~/.gnupg`'.

5 The RecvCmd application

RecvCmd is a java application that runs on a host with ITOS. It accepts STOL commands from SendCmd applications running on remote hosts, decrypts them and passes them to ITOS.

In its default mode, RecvCmd acts as follows:

- An operator must accept every command before it is passed to ITOS.
  This can be changed with the '`-Acceptor`' argument.
- The operator is not able to modify a command when accepting it.
  This can be changed with the '`-arg Acceptor changeCmd`' argument.
- The operator must enter the passphrase used for decryption only the first time it is needed.
  This can be changed with the '`-arg GPG1 promptAll`' argument.
- All replies to SendCmd applications are encrypted and signed.

RecvCmd also:

- Shows the operator the host from which a command was sent and the name used to sign the encrypted version of the command.
- Lets the operator send text messages to SendCmd applications concerning any command submitted.

Invoking RecvCmd
Using RecvCmd
RecvCmd command-line arguments
Example argument files

When RecvCmd runs on a closed network, so SendCmd applications cannot make socket connections to it directly, they send commands to RecvCmd through one or more instances of RelayCmd. There are example argument files to explain this process further.

5.1 Invoking RecvCmd

After installation, RecvCmd can be started by invoking '`runrecv`', usually with no arguments.

'`runrecv`' is a convenience Bourne shell script that sets $CLASSPATH, passes the argument file '`recvargs`' to RecvCmd and runs it in the background.

RecvCmd operation can be modified in two ways:

1. By editing the argument file '`recvargs`'. This is probably the easier method and is described below.
2. By including arguments in the call to '`runrecv`'. Any number of arguments can be given. Type "runrecv -help" or "runrecv ?" for a list of arguments recognized. Those arguments are explained in more detail in the command-line arguments section.

The rest of this page describes the contents of '`recvargs`'. There are also example argument files.

The argument file '`recvargs`' contains the command-line arguments most likely to be useful. Any text following '#' is a comment. The arguments are:

1. -arg ItosComm stolfifo <fifo path>
   This argument is required. As part of installation, "<fifo path>" must be replaced with the STOL fifo path.
2. -arg GPG1 local itos-user
   Specifies the name used to sign outgoing messages and to decrypt incoming messages. '`itos-user`' is correct while using the test keys supplied with the applications, but must be changed when new keys are generated. The user name in this argument should match that in SendCmd's '`-arg GPG1 remote`' argument. If the new name contains blanks, it must be enclosed in single or double quotes.
   If gpg is not used for encryption, this argument should be commented-out.
3. -arg GPG1 gpgpath <path>
   Specifies the path used to invoke gpg. Default is "gpg". The INSTALL script sets this value.
4. -arg GPG1 keydir <directory>
   Specifies the directory where gpg keyring files reside. The INSTALL script sets this value.
5. #-arg GPG1 sign false
   If this argument is uncommented, outgoing messages are not signed.
6. #-arg GPG1 promptAll true
   By default, an operator is required to enter the local user's passphrase only once during an invocation of RecvCmd. If this argument is uncommented, The operator will be queried for the passphrase every time a message is received and, if messages are signed, every time a message is sent.
7. #-arg GPG1 tempdir <directory>
   Specifies where temporary files created during encryption and decryption are put. Default is "/tmp/".
8. #-localPort <port>
   Specifies the server port opened on the local host to receive messages from SendCmd applications. Default is 6100.
   This argument is used slightly differently when RecvCmd connects to a RelayCmd application. See the "SendComm" arguments below and the example argument files.
9. #-logdir <directory>
   Specifies where the log file is put. Default is "/tmp/". To change that, uncomment this argument and replace "<directory>" with a directory path. Log files are never removed or truncated, so it is up to system administrators to delete old files. See "-Log" below for a way to stop log file creation.
10. #-logname <filename>
    Specifies the log file name. Default is '`RecvCmd-mm-dd.log`', where "dd" is the date, "mm" is the number of the month.
11. #-Log NoLog
    Uncommenting this argument will stop any log file from being created.
12. #-arg RecvCmdDB defaultExpiration <min>
    Specifies how long a default command remains in the database before it expires and cannot be accepted. Default is 180 minutes, 3 hours.

Note that this interval applies only to commands whose expiration time is set to "default", not to those with explicit expiration intervals or to those that never expire.

13. #-arg RecvCmdDB delete <days>
    Specifies how many days a command remains in the database after being accepted or rejected. During this time, SendCmd clients can ping to find out what happened to it. Default is 2 days.
14. #-arg RecvCmdDB dbfile <path>
    Specifies which file the database is saved in. Default is '`/tmp/remcmd.recvdb`'.
15. #-arg Acceptor changeCmd true
    Uncommenting this argument permits the operator to modify commands before accepting them.
16. #-Acceptor AcceptAll
    Substitutes an alternate Acceptor class that immediately accepts all commands without operator intervention.
17. #-Decryptor NoDecryption
    #-Encryptor NoEncryption
    Uncommenting these arguments means commands are sent from RecvCmd to SendCmd unencrypted and received commands are not decrypted. This is the preferred way to send commands on a closed network. If these arguments are uncommented, all arguments above that mention GPG1 should be commented-out.
18. #-SendComm RelayComm1
    #-arg SendComm server false
    #-arg SendComm serverHost <host name>
    These three arguments are used when RecvCmd connects to SendCmd applications through a RelayCmd application.
    The first argument is simply uncommented.
    The second argument specifies whether RecvCmd or RelayCmd is the server on their socket connection.
    The fourth argument names the host where RelayCmd is running.
    In addition, the "-localPort" argument, listed above, should give the port opened by the server on this connection whether this application or RelayCmd is the server.
19. #-RecvCmdDB <class>
    This argument substitutes an alternate database class. Default is RecvDB1. There are no alternates at this time.
20. #-ItosComm <class>
    Substitutes an alternate ItosComm class, which sends commands to ITOS. Default is class ItosComm1. There are no alternates at this time.

5.2 Using RecvCmd

The RecvCmd frame contains two buttons:

1. Exit
   Shuts the application down.

2. To the back
   Puts this frame below any other windows on the screen. When a new command arrives, the frame will pop to the top of the window stack.

The RecvCmd frame also contains two scrolled windows.

The lower window displays messages when noteworthy events occur. For example, during startup, messages are printed that name the log file and the port where commands are being accepted.

The upper window lists all commands, along with the time each was received. The most recent command is put at the top of the window. A command's background color tells something about its status:

- A light gray background means the command has been accepted.
- A dark gray background means the command has been rejected.
- A green background means the command has not been accepted or rejected yet.
  If a command is accepted but cannot be sent to ITOS, its background remains green. This happens, for example, if a command is accepted when ITOS is not running.

Clicking on a command in the upper window brings up a frame containing:

- The text of the command
  By default, this text cannot be modified, but if the startup argument
  ```
  -arg Acceptor changeCmd true
  ```
  was used, the text can be changed before the command is accepted.
- All messages referring to the command.
- An "Accept" button
  Clicking this button sends the command to ITOS. This button is deactivated after the command is rejected or successfully sent to ITOS.
- A "Reject" button
  Clicking this button sends a rejection message to the command's creator. This button is deactivated after the command is rejected or successfully sent to ITOS.
- A "Reply" button
  This button is always active and brings up an input window. Any text typed in that input window is sent to the command's creator. This could be used, for example, to explain a rejection.
- A "Delete" button
  This button becomes active only after a command has been accepted or rejected. It removes the command from the command list.
- A "Cancel" button
  Makes the window disappear, but does not affect the command.

When a command is accepted, the following normally happen:

1. The messages
   ```
   <command ID> was accepted.
   <command ID> has been sent to ITOS.
   ```

are printed in RecvCmd's lower window.

2. Any ITOS event window displaying STOL events should display the message:

```
STOL_ECHO: FIFO: <command text>
```

If it seems an accepted command has not been executed by ITOS, check whether the messages described above were printed. If they were, look for STOL_ERROR messages in the ITOS events window referring to the command.

5.3 RecvCmd command-line arguments

It is often more convient to edit the argument file '`recvargs`' than to type command-line arguments.

One argument is **always required** by RecvCmd:

```
-arg ItosComm stolfifo <fifo path>
```

where "<fifo path>" is replaced by the path to the STOL fifo.

Optional arguments:

-localport <number>
: specifies the server port opened for use by SendCmd clients. Default = 6100

-logdir <directory>
: Specifies where the log file is put. Default is '`/tmp/`'.

-logname <filename>
: Specifies the log file name. Default is '`RecvCmd-mm-dd.log`' where mm is the number of the month and dd is the date.

-Log <class>
: Specifies the Log class, which writes the log file. Default is class Log1. Class NoLog does not create a log file.

-arg Acceptor changeCmd <true/false>
: Specifies whether the operator can change a command before sending it to ITOS.

-arg RecvCmdDB defaultExpiration <min>
: Specifies how long a command remains in the database before it expires and cannot be accepted. Default is 180 minutes.
: Note that this interval applies only to commands whose expiration time is set to "default", not to those with explicit expiration intervals or to those that never expire.

-arg RecvCmdDB delete <days>
: Specifies how many days a command remains in the database after being accepted or rejected. During this time, SendCmd clients can ping to find out what happened to it. Default is 2 days.

-arg RecvCmdDB dbfile <path>
: Specifies which file the database is saved in. Default is '`/tmp/remcmd.recvdb`'.

-Decryptor <class>
: Specifies the Decryptor class, which decrypts incoming messages. Default is class GPG1. An alternate is class NoDecryption which does no decryption.

-Encryptor <class>
: Specifies the Encryptor class, which encrypts and signs outgoing messages. Default is class GPG1. An alternate is class NoEncryption, which does no ecnryption.

-Acceptor <class>
: Specifies the Acceptor class, which lets an operator examine commands from SendCmd applications before they are sent to Itos. Default is class Acceptor1. An alternate is class AcceptAll, which immediately sends all commands to STOL with no operator intervention.

-ItosComm <class>
: Specifies the ItosComm class, which passes commands to STOL. Default is class ItosComm1. There are no alternates at this time.

-RecvCmdDB <class>
: Specifies the RecvCmdDB class, which maintains the database of active commands. Default is class RecvDB1. There are no alternates at this time.

The following commands can be used when GPG1 is used for encryption and decryption.

-arg GPG1 sign <true/false>
: Specifies whether outgoing messages are signed. Default is "true".

-arg GPG1 promptAll <true/false>
: Specifies whether the operator is prompted for the local user's passphrase every time an incoming message is decrypted or an outgoing message signed. If false, the operator is prompted only the first time the passphrase is needed. Default is "false"

-arg GPG1 tempdir <path>
: Specifies where temporary files are put. Default is '`/tmp/`'.

-arg GPG1 gpgpath <path>
: Specifies the path to the gpg executable. The default is "gpg", so this argument is necessary only if gpg's directory is not in the $PATH environment variable.

-arg GPG1 keydir <path>
: Specifies the directory containing GPG encryption keys. The default is '`~/.gnupg`'.

The following commands can be used when RecvCmd communicates with SendCmd applications through RelayCmd.

-SendComm <class>
: Specifies the SendComm class, which communicates with all SendCmd applications. If RecvCmd is communicating through RelayCmd, class RelayComm1 should be used.

-arg SendComm server <true/false>
: This parameter **must** be set. It specifies whether this object is socket server or client.

-arg SendComm serverHost <host name>
: This parameter must be set only when RecvCmd is the client.

6 The RelayCmd application

RelayCmd is a java application that relays ITOS commands between a RecvCmd application on a closed network and SendCmd applications on the open internet. For example, RecvCmd might be allowed to make a socket connection out through a firewall to a server socket opened by RelayCmd. SendCmd applications would send requests to RelayCmd, which would relay them to RecvCmd.

There may be situations when ITOS commands must be routed through more than one machine. Any number of RelayCmd applications can be linked to accomplish that.

By default, RelayCmd runs in the background and does not create log files, but during debugging it can display event messages in a window and can write them to a log file.

RecvCmd or any RelayCmd in chain can be shut down and restarted independently. Sockets with running processes will automatically be re-established on restart.

Invoking RelayCmd
Sample argument files

6.1 Invoking RelayCmd

RelayCmd can be started by invoking '`runrelay`', usually with no arguments.

'`runrelay`' is a convenience Bourne shell script that sets $CLASSPATH, passes the argument file '`relayargs`' to RelayCmd and runs it in the background. The rest of this page describes the contents of '`relayargs`'. There are also some example argument files.

The argument file '`relayargs`' contains command-line arguments most likely to be useful. Any text following the character '#' is considered a comment. The arguments are:

- #-createFrame true
  Uncommenting this argument makes RelayCmd display messages in a window. This can be useful during debugging.
- #-Log Log1
  Uncommenting this argument makes RelayCmd create a log file.
- #-logdir <directory>
  Specifies where the log file is put. Default is "/tmp/". This is useful only when "-Log Log1" is uncommented.
- #-logname <filename>
  Specifies the log file name. Default is '`RelayCmd-mm-dd.log`' where "dd" is the date, "mm" is the number of the month. This is useful only when "-Log Log1" is uncommented.

The following arguments refer to communications with the application on the ITOS side of RelayCmd. That can be RecvCmd or another instance of RelayCmd.

- -arg ItosComm server true
  Specifies whether this application is the server or client on the socket. The application on the other end of the socket should use the opposite setting.

- -arg ItosComm serverPort 6101
  Sets the server port number used for the socket, regardless of which application is the server.
- #-arg ItosComm serverHost <host name>
  If this application is not the socket server, this argument is needed to tell it the name of the host where the other application is running.

The following arguments refer to communications with the application on the SendCmd side of this application. That can be another instance of RelayCmd or a group of SendCmd applications.

- #-SendComm RelayComm1
  Uses RelayComm1 to send outgoing messages instead of the default, SendComm1. This should be uncommented when there is another RelayCmd application between this application and SendCmd clients.
- #-arg SendComm server <true/false>
  This argument is used only when RelayComm1 is used as the SendComm class. It Specifies whether this application or the other RelayCmd is the server on their socket connection. This argument is **required** when RelayComm1 is used.
- #-arg SendComm serverPort <number>
  Specifies the port number of the server socket in the connection between this application and the next application on the SendCmd side. If there is a RelayCmd between this application and SendCmd clients, and this application is not the server, this argument specifies the server port opened by the other RelayCmd. Default is 6100.
- #-arg SendComm serverHost <host name>
  This argument is used only when RelayComm1 is used as the SendComm class. If this application is not the server, this argument is needed to tell it the name of the host where the other RelayCmd is running.

6.2 Example sets of argument files

The examples below concentrate on arguments that must be coordinated between applications. For other arguments used in argument files, see:
SendCmd argument files
RecvCmd argument files
RelayCmd argument files

6.2.1 Example 1: Direct connection between SendCmd and RecvCmd

Suppose RecvCmd is running on a machine named "california", which is accessible from the open internet. Here are some lines from its '`recvargs`' file and from '`sendargs`' of SendCmd applications on other machines that communicate with it:

```
recvargs on California          sendargs
----------------------          -----------------------
-arg GPG1 local itos-user       -RecvCmd california 6100
```

```
#-SendComm SendComm1                -arg GPG1 local remote-user
#-localPort 6100                    -arg GPG1 remote itos-user
```

The host name "california" was entered manually in 'sendargs' during installation. All other values are unchanged from their original settings. In 'recvargs' the "localPort" and "SendComm" lines remain commented-out and are shown here only to show their default values and because they will be uncommented in later examples.

Notice that the port number in "-RecvCmd california 6100" matches the default port in recvargs. If that port was changed on california, it would also have to be changed in all 'sendargs' files.

Also notice that the "GPG1 local" argument in recvargs must match the "GPG1 remote" argument in all sendargs files.

6.2.2 Example 2: One RelayCmd between SendCmd and RecvCmd

Suppose RecvCmd, again running on california, is not accessible from the open internet. There is a gateway machine, named "oregon", which is accessible from the open internet and processes running on california can be given permission to connect to server sockets opened on oregon.

RelayCmd will run on oregon and read command-line arguments from 'relayargs'. RelayCmd will open a server socket on port 6101 for its connection to RecvCmd. It will open a server socket on port 6100 to receive ITOS commands from SendCmd applications. The following argument files make that happen:

```
recvargs on California           relayargs on oregon
----------------------           -------------------
-arg GPG1 local itos-user        #-ItosComm <class>
-SendComm RelayComm1             -arg ItosComm server true
-arg SendComm server false       -arg ItosComm serverPort 6101
-arg SendComm serverHost oregon  #-arg SendComm serverPort <number>
-localPort 6101

sendargs
----------------------
-RecvCmd oregon 6100
-arg GPG1 local remote-user
-arg GPG1 remote itos-user
```

In 'recvargs':

- Argument "SendComm RelayComm1" starts a class that knows how to communicate with RelayCmd rather than the default class, which communicates with SendCmd applications.
- Argument "SendComm server false" tells RecvCmd it will not be the server on this connection. It complements "ItosComm server true" in 'relayargs'.
- Argument "SendComm serverHost oregon" tells RecvCmd that the server socket is being opened on host "oregon".

- Argument "localPort" gives the port number of the server socket and matches the "ItosComm serverPort 6101" argument in '`relayargs`'.

In '`relayArgs`', all "ItosComm" arguments refer to the connection between RelayCmd and the application to its ITOS side. In this case, that is RecvCmd.

- Argument "-ItosComm" remains commented-out because its default, RelayComm1, is the correct class to communicate with the RelayComm1 in RecvCmd.
- Argument "ItosComm server true" tells RelayCmd it will be the server on this connection. It complements the "SendComm server false" in '`recvargs`'.
- Argument "ItosComm serverPort 6101" tells RelayCmd to use port 6101 for its server socket. It matches the "localPort 6101" argument in '`recvargs`'.
- Argument "SendComm serverPort" specifies the port that RelayCmd will open to accept ITOS commands from SendCmd applications. The default port, 6100, is satisfactory in this case, so this argument is commented-out.

In '`sendargs`':

- Argument "RecvCmd oregon 6100" tells SendCmd to send commands to port 6100 on "oregon", where RelayCmd is running.

6.2.3 Example 3: Two RelayCmds between SendCmd and RecvCmd

Suppose RecvCmd, again running on california, must relay commands through a RelayCmd running on oregon to a second relayCmd running on a machine named "washington", which is accessible from the open internet.

As in example 2, RecvCmd is the client on a socket to RelayCmd on oregon using port 6101.
RelayCmd on oregon will be the client on a socket to RelayCmd on washington, also using port 6101. SendCmd applications will send ITOS commands to port 6100 on washington.

```
recvargs on California           relayargs on oregon
----------------------           -------------------
-arg GPG1 local itos-user        #-ItosComm <class>
-SendComm RelayComm1             -arg ItosComm server true
-arg SendComm server false       -arg ItosComm serverPort 6101
-arg SendComm serverHost oregon  -SendComm RelayComm1
-localPort 6101                  -arg SendComm server false
 -arg SendComm serverPort 6101
                                 -arg SendComm serverHost washington

relayargs on washington          sendargs
-----------------------          ---------------------
-ItosComm RelayComm1             -RecvCmd washington 6100
-arg ItosComm server true        -arg GPG1 local remote-user
-arg ItosComm serverPort 6101    -arg GPG1 remote itos-user
```

'`recvargs`' is unchanged from example 2.

In file ‘`relayargs`’ on oregon, the "ItosComm" arguments describe the connection to RecvCmd and are unchanged from example 2. The "SendComm" arguments describe the connection to RelayCmd on washington:

– Argument "SendComm RelayComm1" starts a class that knows how to communicate with RelayCmd rather than the default class, which communicates with SendCmd applications.
– Argument "SendComm server false" tells RelayCmd it will not be the server on this connection. It complements "ItosComm server true" in ‘`relayargs`’ on washington.
– Argument "SendComm serverPort 6101" tells RelayCmd the port number to be opened by the server.
– Argument "SendComm serverHost washington" tells RelayCmd that the server socket is being opened on host "washington".

In file ‘`relayargs`’ on washington, the "ItosComm" arguments describe the connection to RelayCmd on oregon:

– Argument "ItosComm RelayComm1" starts a RelayComm1 class to communicate with RelayCmd on oregon. This matches the "SendComm RelayComm1" argument in ‘`relayargs`’ on oregon.
– Argument "arg ItosComm server true" tells RelayCmd it will be the server on this connection. It complements the "SendComm server false" in ‘`relayargs`’ on oregon.
– Argument "ItosComm serverPort 6101" matches the "SendComm serverPort 6101" in ‘`relayargs`’ on oregon.

All SendComm arguments on washington have been commented-out because RelayCmd on washington communicates directly with SendCmd applications so default settings are correct.

All ‘`sendargs`’ must be modified to set the host in the "RecvCmd" argument to "washington".

7 Encryption/Decryption operations

Encryption is a complex topic, beyond the scope of this document. The following links give some basic information:

-
-
-
-
-

The following contains a user's manual for gpg, the cryptographic package used here:

Two encryption keys are included with the ITOS distribution files for use during testing:

- The public key for user "itos-user" is used to encrypt messages sent to the RecvCmd application. RecvCmd uses the private key for this user to decrypt messages, so it will ask for the key's passphrase, which is "itos-user".
  The private key for this user is also used to sign messages sent by RecvCmd.
- The public key for user "remote-user" is used to encrypt messages sent to SendCmd applications, so SendCmd will ask for the key's passphrase, which is "remote-user".
  The private key for this user is also used to sign messages sent by SendCmd.

Before normal operations begin, custom keys should be generated to replace these test keys. This involves four steps:

1. Remove the test keys. This is not necessary, but insures that the test keys are not used inadvertently. This can be done with
   ```
   rm gpg/*.gpg*
   ```
2. Generate new keys. More about this below.
3. Distribute the new keys to all sites. One secure way to do this is to copy files gpg/pubring.gpg and gpg/secring.gpg to a floppy and send the floppy to all sites.
4. If the new keys have different names from the test keys, argument files '`sendargs`' and/or '`recvargs`' must be modified. Both contain lines like:
   ```
   -arg GPG1 local remote-user
   -arg GPG1 remote itos-user
   ```
   Strings "remote-user" and "itos-user" should be changed to the new user names. If the names contain blanks, they must be enclosed in single or double quotation marks.

The following command can be used to generate a new key:

```
rungpg --gen-key
```

That command will ask several questions. An excellent place to learn about the issues involved in generating keys and other aspects of using gpg is

Script "rungpg" is provided to remove two steps in executing gpg commands. It sets the environment variable $GNUPGHOME so pgp looks in the correct directory for key files and it uses a complete path when invoking gpg so it is not necessary to have gpg in your $PATH. Other than that, running gpg commands with rungpg is exactly like invoking gpg directly.

You will create at least two keys, one to decrypt messages sent to RecvCmd and the other to decrypt messages sent to SendCmd applications. Different SendCmd applications may use different keys and at least the public parts of all those keys must be available to RecvCmd.

Some other useful operations are:

-
  ```
  rungpg --list-keys
  rungpg --list-public-keys
  rungpg --list-secret-keys
  ```
  List keys currently on the keyring.

-
  ```
  rungpg --delete-key
  ```
  Deletes a key from the ring.

-
  ```
  rungpg -u itos-user -r remote-user --sign -o file2 --encrypt file1
  ```
  Encrypts the file '`file1`' and puts the encrypted version in file '`file2`'. The encrypted version is signed by itos-user and must be decrypted with remote-user's private key. You will be asked for itos-user's passphrase for the signiture.

-
  ```
  rungpg -u remote-user -o file3 --decrypt file2
  ```
  Decrypts file '`file2`' created above and puts the plain text version in '`file3`'. It should be identical to '`file1`'. You will be asked for remote-user's passphrase for the decryption.

-
  ```
  rungpg --edit-key remote-user
  ```
  Allows you to change a user's properties. The user in this case is remote-user. The prompt "Command>" will appear. Type "help" for a list of commands.
  The command "passwd" allows you to change the password for this user. You will be asked to enter the user's current password and then the new password twice.